Tools - Dependency Injector

Purpose

The purpose of this tool is to help find missing dependencies in the configuration of a J2EE application, based on the warnings found in an analysis log file. It can store the results into a CSV file, or directly inject the missing dependencies into the management base.

In 7.3 and starting from 8.2.8, dependencies are not mandatory for links to be drawn by the J2EE analyser. So the tool is useful for versions betweeen 8.0/0 and 8.2.7 included.

This tool is a helper and does not replace human involvement in delivery/packaging review. In cases where objects belong to several analysis units, it might not choose the proper one.

Dependency injector is based on two types of warnings.

Warnings used by dependency_injector

```
class ...  is not visible - one dependency is missing
can not resolve .... as type in package .... from package ......
```

Applicable in CAST Version

Release	Yes/No
8.3.x	✗
8.2.x	✓ (x <=7)
8.1.x	✓
8.0.x	✓
7.3.x	✗

Applicable RDBMS

RDBMS	Yes/No
Oracle Server	✓
Microsoft SQL Server	✓
CSS2	✓

Prerequisites

Follow the below prerequisites before running the tool:

CAST-AIP

dependency_injector must be installed on the same machine as CAST-AIP.

dependency_injector.py is a Python script, so you need to launch it with Python 3.4 embedded in Cast AIP.

Consistency

The analysis log file and the management base must be CONSISTENT for optimized results. You should always run the dependency_injector on the latest analysis log file.

No overlapping web analysis units

There must be no overlapping web analysis units in your configuration.

dependency_injector relies on the analysis units the objects quoted in the warnings belong to. So it first performs a check on overlapping web analysis units and proceeds only if it finds none.

Scope

Warnings triggered by classes located in JAR files should be ignored

Warnings triggered by classes located in JAR files cannot be fixed, as adding dependencies will not make them disappear. However, classes in JAR files are EXTERNAL objects,
so the analyzer will not draw any links starting from them anyway. So these warnings have no consequences on links and transactions.

Warnings whose source objects belong to several analysis units may not be solved

If the source of the warning is implied in a warning like the following:

```
2017-11-28 15:04:23,340 - __main__ - WARNING - Source object [S:\Sources\APP\Analyzed\APP_UTILITIES\WebContent\WEB-INF\struts-config.xml] belongs to several analysis units, please review pakaging / configuration
2017-11-28 15:04:23,340 - __main__ - WARNING - Source object [S:\Sources\APP\Analyzed\APP_UTILITIES\WebContent\WEB-INF\struts-config.xml] belongs to : XXXXXX (id=58938, type=ECLIPSE), YYYYYY (id=151626, type=USER_DEFINED)
2017-11-28 15:04:23,340 - __main__ - WARNING - Arbitrarily chose AU YYYYYYfor object [S:\Sources\APP\Analyzed\APP_UTILITIES\WebContent\WEB-INF\struts-config.xml]
```

Then it means that dependency-injector is unable to determine for sure which analysis unit the object belongs.
Anyway your configuration should be fixed.

Usage

This tool must be used after a first analysis, as it takes an analysis log file for input.
A first output will give the missing dependencies based on the warnings found in the first log.
Add the missing dependencies, then re-run analysis.

Adding some dependencies helps resolution go further.... and new warnings can appear. Most big applications will require to run the {tool + new analysis} twice or three times
in order to suppress every warning.

You can stop iteration when the CSV output file for one run is identical to the previous one, OR when dependency-injector says : "0 dependencies were inserted"
After a maximum of 2 or 3 runs, all your warnings should be either explained (see Scope) or removed.

Launch Dependency Injector

Usage

dependency_injector.py is a Python script, so you need to launch it with Pyhon 3.4 embedded in Cast AIP.

It must be used with one of the following sub-commands :

- **compute :** only compute the missing dependencies and save the results into a CSV file
- **inject :** takes as input a CDV file generated by the compute sub-command and injects the dependencies into the management base
- **all** : computes the missing dependencies, optionally saves the results into a CSV file, then injects the dependencies into the management base

Mandatory arguments :

- **host** : the hostname of the CSS server where the management base is installed
- **mb** : the name of the management base
- **kb:** the name of the knowlegde base
- **log :** dependency injector log file (not the analysis log)

Mandatory arguments for sub-command compute :

- **report :** the output CSV file where the tool will store the computed missing dependencies
- **input :** the input analysis log file

Mandatory arguments for sub-command all (note that argument **report** is optional if you use option **all)**:

- **input** : the input analysis log file

Optional arguments :

- **port** : the port of the CSS server where the management base is installed (default : 2280)
- **user** : the user name used to connect to the CSS server (default : operator)
- **password** : the password used to connect to the CSS server (default : CastAIP)
- **verbose :** more details are given in the log file and on the standard output (default : False)

Open a CMD window, and navigate to the folder where you installed classpath_checker. Type one of the following command-lines :

Command-line

```
<INSTALL_FOLDER>\ThirdParty\Python34\python.exe dependency_injector.py compute -host <CSS SERVER> -mb
<MANAGEMENT_BASE> -kb <KNOWLEDGE_BASE> -i <ANALYSIS_LOG_FILE> -r <REPORT> -log <DEPENDENCY_INJECTOR_LOG>

<INSTALL_FOLDER>\ThirdParty\Python34\python.exe dependency_injector.py inject -host <CSS SERVER> -mb
<MANAGEMENT_BASE> -kb <KNOWLEDGE_BASE> -r <REPORT> -log <DEPENDENCY_INJECTOR_LOG>

<INSTALL_FOLDER>\ThirdParty\Python34\python.exe dependency_injector.py all -host <CSS SERVER> -mb
<MANAGEMENT_BASE> -kb <KNOWLEDGE_BASE> -i <ANALYSIS_LOG_FILE> -r <REPORT> -log <DEPENDENCY_INJECTOR_LOG>
```

Output File

If you use compute subcommand, then the ouput will be CSV File with the following columns :

Source_ID	Source_Name	Source Type	Target_ID	Target_Name	Target Type

Source_ID is the ID of the source analysis unit

Source_Name is the name of the source analysis unit

Source_Type is the type of discoverer that created the source analysis unit : for example, **MAVEN**, **ECLIPSE**, **J2EE_FILE_DISCOVERER**, or **User-Defined** if it was created by hand. This piece of information can help you understand why the dependency was not discovered.

Target_ID is the ID of the target analysis unit

Target_Name is the name of the target analysis unit

Target_Type is the type of discoverer that created the target analysis unit : for example, **MAVEN**, **ECLIPSE**, **J2EE_FILE_DISCOVERER**, or **User Defined** if it was created by hand. This piece of information can help you understand why the dependency was not discovered.

Command-line output

Sub-command compute:

If the name of the analysis log file does not match one of the entries for column **execution_log** of table **cms_j2ee_analysis** in Management Base, then you have the following warning message. You are allowed to proceed by typing yes, because the file could have been copied and renamed.

Log-MB consistency

```
C:\temp\Tools\DependencyInjector>"C:\Flat Service Packs\8.2.7\ThirdParty\Python34\python.exe"
dependency_injector.py compute -host localhost -mb app_mngt -kb app_local -i C:\temp\10341\HELLO.castlog -r C:
\temp\10341\TKB.csv -log C:\temp\10341\TKB.log


Castlog file HELLO.castlog does not seem to be the last analysis log file
The last log file should be in this list, unless it was renamed :
[My Source file based execution unit_2856-20180209203535.castlog]

Using the wrong log may lead to inconsistent results. Do you want to proceed ? [y/N]
```

If you have overlapping web analysis units in the Management Base, **dependency_injector** will tell you which ones, suggest some action and exit

Overlapping web analysis units

```
Analysis unit Generated (id=127386) is overlapping with analysis unit yyy (id=7361)
S:\Sources\APP\Analyzed
S:\Sources\APP\Analyzed\zzzzz\yyy\.project

Analyis unit Generated (id=127386) is overlapping with analysis unit kkkk-server-jar (id=130297)
S:\Sources\INSTAR\Analyzed
S:\Sources\INSTAR\Analyzed\kkkk-server\pom.xml

Done

Cannot run until all the overlapping web analysis units are fixed, else results will be wrong.
Fix suggestions :
 - Disable web for one of the overlapping analysis units
 - or change the root path of one of the analysis units so that it does not overlap the other one
 - or merge the overlapping analysis units into one unique web analysis unit
```

If no overlap is detected, then **dependency_injector** starts parsing the analysis log file and resolves the source analysis units and the target analysis units. Then it computes the missing dependency graph, taking transitivity into account.
Then saves the results to the output file

Parsing the analysis log file

```
Analyzing file C:\temp\10341\APP-10341.castlog
Found 589 warnings
Found 530 distinct issues at the object level
Resolving 178 source objects...
... to 60 source analysis units
Resolving 279 target objects...
... to 27 target analysis units
Found 98 missing dependencies
Computing missing dependency graph
Ignoring 16 dependencies due to transitivity
Remaining 82 missing dependencies
Saving results to output file C:\temp\10341\TKB.csv

Do not forget to send file C:\temp\10341\TKB_20180227103110.drr to CAST R&D
```

Sub-command inject:

dependency_injector reads the CSV file, installs the necessary Stored Procedures on the Management Base, and then injects the missing dependencies

Reading the CSV file

```
C:\temp\Tools\DependencyInjector>"C:\Flat Service Packs\8.2.7\ThirdParty\Python34\python.exe"
dependency_injector.py inject -host localhost -mb app_mngt -kb app_local -r C:\temp\10341\TKB.csv -log C:
\temp\10341\TKB.log

Reading file C:\temp\10341\TKB.csv
['151626', 'SSAMLoggingEAR', 'USER_DEFINED', '58920', 'SPCUAREJB', 'ECLIPSE']
['151626', 'SSAMLoggingEAR', 'USER_DEFINED', '58937', 'SSAMTextSearchUtilities', 'ECLIPSE']
[...]
Read 82 dependencies
2 dependencies were already present
Installing Tasklog API on sdp_mngt
Installing MAINT_Tasklog_Start_Action.sql
Installing MAINT_Tasklog_End_Action.sql
Installing MAINT_Tasklog_Start_Session.sql
Installing MAINT_Tasklog_End_Session.sql
Installing MAINT_Tasklog_Additional_Information.sql
Installing procedure SUP_CreateRequiredDependencies on app_mngt
Executing procedure SUP_CreateRequiredDependencies on app_mngt
80 dependencies were inserted

Do not forget to send file C:\temp\10341\TKB_20180228100959.drr to CAST R&D
```

Notes/comments

Tickets #10341, 10538, 10540

Related Pages

J2EE - application qualification specifics